

Agentic Coding

The Future of Software Development with AI Agents

About Me

Hi, I'm Armin *mitsuhiko* Ronacher!

Passionate about building teams, products, and Open Source

-  **Creator of Flask, Jinja2** and many other widely-used OSS projects
-  **Decade at Sentry** - first engineer, had hands in a lot of things
-  **Conference Speaker** - sharing knowledge about engineering and life
-  **AI Programming Explorer** - currently diving deep into agentic development

Currently...

I currently don't sleep a lot
Claude Code is catnip for programmers.

What is Agentic Coding?

Agentic coding is a software development process where AI agents actively participate in the coding process:

-  **AI-Powered Assistance** - Intelligent agents that understand context and intent
-  **Interactive Development** - Real-time collaboration between human and AI
-  **Task-Oriented** - Agents that can complete complex, multi-step programming tasks
-  **Context Awareness** - Understanding of codebase structure, patterns, and conventions

Instead of just suggesting code, agents actively participate in planning, implementing, and refining solutions.

Why is this a thing all the sudden?

The Perfect Storm

- **Sonnet 4 & Opus 4** - Breakthrough models trained specifically for tool usage
- **Claude Code** - Demonstrated what's possible with AI-powered development
- **Explosive Growth** - Everyone is now building their own Claude Code-inspired agents
- **Claude Max Subscription** - a flat fee of \$100/\$200 per month

{ Anthropic went all in on programming agents and we all followed.

Cursor vs Claude Code

What really is the difference?

- **yolo** - we are hooked on Claude Code with `--dangerously-skip-permissions`, aka yolo mode
- **the role of the editor** - with Claude Code you review, you program much less
- **a stepping stone** - long term maybe background agents are the destination, we are not there

How does it compare to Cursor's agents and background agents?

- **Agents:** don't go for as long as Claude Code is
- **Background agents:** are a bit too separate (in the cloud), make it harder to optimize and review

What Tools Exist Today?

The screenshot shows the OpenCode CLI interface. At the top, it says "OPENCODE v0.1.150". Below that is a command-line interface with the following text:

```
/help      show help          ctrl+X h
/editor    open editor        ctrl+X e
/models   list models        ctrl+X m
/init     create/update AGENTS.md
/compact  compact the session ctrl+X i
/sessions list sessions      ctrl+X c
```

At the bottom, there's a code editor window with the title "Anthropic Claude Sonnet 4".

The screenshot shows the Claude-YOLO CLI interface. At the top, it says "mitsuhiko at cheetah in ~/Development/minijinja on git:main rust 1.88.0 \$ claude-yolo". Below that is a welcome message:

```
* Welcome to Claude Code!
/help for help, /status for your current setup
cwd: /Users/mitsuhiko/Development/minijinja
```

There's also a tip about creating a todo list and a command-line interface for refactoring:

```
! Tip: Ask Claude to create a todo list when working on complex tasks to track progress and remain on track
> !y "refactor <filepath>"
```

The screenshot shows the Gemini CLI interface. At the top, it says "Gemini - minijinja". Below that is a large, stylized "GEMINI" logo.

There are "Tips for getting started:" and a numbered list:

1. Ask questions, edit files, or run commands.
2. Be specific for the best results.
3. Create GEMINI.md files to customize your interactions with Gemini.
4. /help for more information.

At the bottom, it says "YOLO mode (ctrl + y to toggle)" and "Type your message or @path/to/file".

- **OpenCode** - Open source alternative for AI-powered development
- **Amp** - AI coding assistant platform
- **OpenAI's Codex** - The original AI coding model
- **Gemini CLI** - Google's command-line AI assistant
- *and many more...*

“Making the \$200 Worth It”

This is the top post on the /r/cursor
subreddit right now.
It shows Claude Code.

I use Claude Code as my Terminal

- I ask it to investigate issues and read docs
- I ask it to setup and debug CI for me
- I ask it to configure my machine for me
- I ask it to create and use small tools for me
- I ask it to browse the internet for me
- **I used Claude Code to create this very presentation**

Claude Code has become my intelligent command line that understands context and can perform complex multi-step operations.

Debugging Shell Environment with Claude

My Agentic Coding Recommendations

What I learned so far programming with agents.

1. Language and Ecosystem Choice

Simplicity Helps

- **Simple Languages work best** - Go, PHP, Basic Python, ...
- **Prefer low ecosystem churn** - Stable foundations work best for agent-assisted development
- **Long function names beat namespaces** - agents are good, but don't make them depend on the LSP

For existing code bases, start simple and find ways to wedge yourself in:

- Invest into your dev environment!
- Centralize logging and observability

2. Tool and Workflow Principles

Build Agent-Friendly Tools

- **Create fast, user-friendly tools** - Speed matters for agent iteration
- **Prioritize logging and observability** - Agents need to understand what's happening
- **Protect tools against potential misuse** - Defensive programming
- **Enable quick tool creation and execution** - Reduce friction

Just tell Claude where it should place bespoke tools!

2a. MCP

- **MCP is a way to give Agents more tools**
- MCP is useful for things like playwright (remote control browsers)
- Too much MCP causes faster context rot and rarely works better than command line tools
- For coding agents, I encourage not to use MCP

I get better results creating bespoke tools and run them than to use MCP tools.

3. Conserve Context

It's all about conserving context

- Prevent it from spelunking:
 - Give it tools to effectively navigate the code base
 - Provide ways to tail the last 20 lines of combined logs
- A ``CLAUDE.md`` that's too long is not helpful
- Consider using sub tasks / sub agents to conserve context
- When you need to ``/compact`` you lost

4. Enable Forward Progress

- A broken dev environment can cause back-tracking
- The agent can get confused if the change broke the environment of it came pre-broken
- Tools that provide noisy output negatively contribute to context rot
- Backtracking even when corrected rots the context badly
- **Cool shit:** test caching in go
- **Uncool shit:** rust's test runner does not error when a null-selection was made (accidentally assumes stuff passed)

Practical Things I Learned

Real-world tips from my experience with agentic coding

Tip 1: Unified Logging

Forward Everything Into One Log File

- **Combine `console.log` + server logs + everything else**
- patch ``console.log`` in the browser -> forward to server via API call
- All output streams flow to a single, tailable log file
- Give it a way to log out SQL too!
- Provide a ``make tail-logs`` command for easy access

```
# Example  
make tail-logs # Shows last 50 lines, follows new output
```

Tip 2: Multi-Process Guidance

Give the Agent Context About Process Orchestration

- **Document how multiple processes work together**
- Explain startup order and dependencies
- Provide clear process status commands
- Make it obvious which processes need to be running

Don't make the agent guess how your distributed system works.

Tip 3: Synchronization Points

Complex Systems Need Await Mechanisms

- **Give agents ways to wait for synchronization points**
- Provide health check endpoints
- Create "wait-for-ready" commands
- Make async operations observable

```
from .agentsupport import reached  
reached(point="event-preprocessing-done")
```

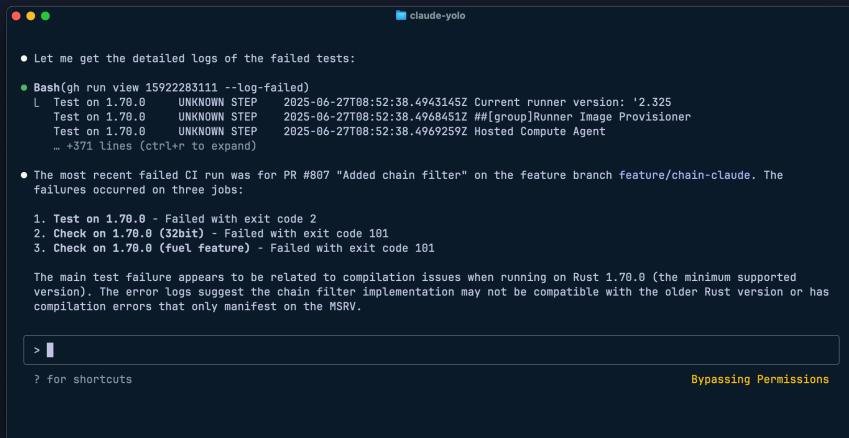
```
make await POINT=event-preprocessing-done # Block until reached or timeout
```

Tell the agent it can do that, and it will happily write sync points into the code to debug.

Tip 4: CI Debugging Access

Claude Code Can Debug Your CI / Debug In CI

- Sometimes your shit only breaks in CI.
- **Give it access to GitHub!**
- Use ``gh`` CLI for pull requests and issues
- Let it create branches and PRs directly, let it



The screenshot shows a terminal window titled "claude-yolo". The window contains the following text:

- Let me get the detailed logs of the failed tests:
- Bash(gh run view 15922283111 --log-failed)
L Test on 1.70.0 UNKNOWN STEP 2025-06-27T08:52:38.4943145Z Current runner version: '2.325
Test on 1.70.0 UNKNOWN STEP 2025-06-27T08:52:38.4968451Z ##[group]Runner Image Provisioner
Test on 1.70.0 UNKNOWN STEP 2025-06-27T08:52:38.4969259Z Hosted Compute Agent
... +371 lines (ctrl+u to expand)
- The most recent failed CI run was for PR #807 "Added chain filter" on the feature branch feature/chain-claude. The failures occurred on three jobs:
 1. Test on 1.70.0 - Failed with exit code 2
 2. Check on 1.70.0 (32bit) - Failed with exit code 101
 3. Check on 1.70.0 (fuel feature) - Failed with exit code 101

The main test failure appears to be related to compilation issues when running on Rust 1.70.0 (the minimum supported version). The error logs suggest the chain filter implementation may not be compatible with the older Rust version or has compilation errors that only manifest on the MSRV.

> [redacted]
? for shortcuts Bypassing Permissions

Weird Shit I Did

Just to get you an idea how deep the rabbit hole gets.

Use Agent 1 with Agent 2

- I instruct my Claude to use Gemini CLI when it needs to summarize
- Why? Because Gemini has more context (but it sucks at tooling)

- * When analyzing large codebases or multiple files that might exceed context limits, use the Gemini CLI with its massive context window.
- * Use `gemini -p` when:
 - Analyzing entire codebases or large directories
 - Comparing multiple large files
 - Need to understand project-wide patterns or architecture
 - Checking for the presence of certain coding patterns across the entire codebase

Examples:

```
gemini -p "@src/main.py Explain this file's purpose and structure"
gemini -p "@src/ Summarize the architecture of this codebase"
gemini -p "@src/ Are there any React hooks that handle WebSocket connections? List them with file paths"
```

Scraped Videos

- The Video on an earlier slide? I asked Claude to download it and put into my presentation:

I want a new slide at the end that embeds the video from this page. we might have to download it first:
https://www.reddit.com/r/cursor/comments/1lmhm5x/idk_how_you_guys_are_using_claude_code_but_im/

- Bash(uvx yt-dlp "https://www.reddit.com/r/cursor/comments/1lmhm5x/idk_how_you_guys_are_using_claude_code_but_im/" --info-json)
L Error: Usage: yt-dlp [OPTIONS] URL [URL...]

yt-dlp: error: no such option: --info-json
- Bash(uvx yt-dlp "https://www.reddit.com/r/cursor/comments/1lmhm5x/idk_how_you_guys_are_using_claude_code_but_im/" --write-info-json
--skip-download)
L [Reddit] Extracting URL: https://www.reddit.com/r/cursor/comments/1lmhm5x/idk_how_you_guys_are_using_claude_code_but_im/
[Reddit] 1lmhm5x: Downloading JSON metadata
[Reddit] 1lmhm5x: Downloading m3u8 information
... +4 lines (ctrl+r to expand)
- Read(Idk how you guys are using Claude code but im making my 200 usd worth it [113pqwijmm9f1].info.json)
L Read 1 line (ctrl+r to expand)
- Perfect! Now I can see the video details. It's a 28-second video titled "Idk how you guys are using Claude code but im making my 200 usd worth it" from Reddit. Let me download the actual video file:
- Bash(uvx yt-dlp "https://www.reddit.com/r/cursor/comments/1lmhm5x/idk_how_you_guys_are_using_claude_code_but_im/" -o
"claude-code-demo.%exts")
L [Reddit] Extracting URL: https://www.reddit.com/r/cursor/comments/1lmhm5x/idk_how_you_guys_are_using_claude_code_but_im/
[Reddit] 1lmhm5x: Downloading JSON metadata
[Reddit] 1lmhm5x: Downloading m3u8 information
... +11 lines (ctrl+r to expand)
- Great! Now let me update the todos and then read the current end of the slides file to see where to add the new slide:

Sell Shit on The Internet

- Playwright is good enough to put things on willhaben (Craigslist)
- Have a profile that's signed in
- It will gladly do the laborous task for you from finding price, to writing description.
- Just put the pictures somewhere and tell it to do the rest

Reconfigure Tools

- Can't figure out how stuff works? The LLM knows

```
> i want this to happen automatically globally. is there a setting?
```

- Yes, there are two settings to make this automatic globally:
 - **Bash**(git config --global push.autoSetupRemote true)
└ (No content)
 - **Bash**(git config --global pull.rebase false)
└ (No content)
- The first command (push.autoSetupRemote true) automatically sets up upstream tracking when you push a new branch. The second prevents Git from asking about pull strategy every time.

Now when you create new branches and push them, Git will automatically set up tracking so you can use `git pull` without arguments.

What will the future bring?

We are very early, but you can feel the shift.